REMARKS

In the Official Action mailed on **20 December 2007**, the Examiner reviewed claims 1-4, 6-16, and 18-25. Examiner objected to claim 13. Examiner provisionally rejected claims 1-4, 6-16 and 18-25 on the ground of non-statutory obviousness-type double patenting as being unpatentable over claim 1, 3, 5-13, 15, and 17-25 of co-pending U.S. Application No. 10/637,166. Examiner rejected claims 1-3, 6-15, and 18-25 under 35 U.S.C. § 103(a) based on Moss et al ("*Transactional Memory: Architectural Support for Lock-Free Data Structures*," hereinafter "Moss") in view of Oplinger et al ("*Enhancing Software Reliability with Speculative Threads*," hereinafter "Oplinger").

Claim Objections

Examiner objected to claim 13 for not concluding with a period. Applicant has amended claim 13 to include a period. Applicant therefore respectfully requests the withdrawal of the objection.

Obviousness-Type Double Patenting Rejection

The Examiner provisionally rejected claims 1-4, 6-16, and 18-25 on the ground of non-statutory obviousness-type double patenting as being unpatentable over claims 1, 3, 5-13, and 17-25 of co-pending Application No. 10/637,166, in view of Rajwar and Moss.

Applicant has included a Terminal Disclaimer with the instant Office Action Response to overcome the Double Patenting Rejection. Accordingly, Applicant respectfully requests the withdrawal of the Double Patenting Rejection.

Rejections under 35 U.S.C. § 103(a)

Examiner rejected claims 1-3, 6-15, and 18-25 under 35 U.S.C. § 103(a) as being unpatentable over Moss in view of Oplinger. Examiner argues:

> "Oplinger teaches a transactional programming model in which his abort/fail instruction not only eliminates the speculative state, but also branches the program to an address that was previously set by a TRY instruction (Section 3.2). The advantage of having an instruction is being able to go to an error handling case in the event of an abort (see Section 3.2, the second code example, where the system jumps to an error message on an abort), giving the programmer more control over the execution and troubleshooting of his program (see Office Action, page 5)."

Applicant respectfully disagrees. Moss and Oplinger are fundamentally distinct from embodiments of the present invention because the combination of Moss and Oplinger discloses an ABORT instruction that is limited to aborting transactional execution and branching to a location set by the TRY instruction.

Moss discloses a very simple version of an ABORT instruction that is limited to **terminating transactional execution** and discarding "all updates to the write set" (see Moss, section 2.1). On the other hand, Oplinger discloses an ABORT instruction that is confined to **aborting the transaction**, discarding the speculative state, and setting the program counter to the address specified by the last TRY instruction (see Oplinger, section 3.2). Hence, in the broadest reading, the combination of Moss and Oplinger discloses an ABORT instruction which jumps to a target provided by a TRY instruction (as disclosed by Oplinger).

In contrast, embodiments of the present invention provide a FAIL instruction that provides a programmer more control over program execution. Using the FAIL instruction, the programmer can **update the state information of the processor** with information about the failure, and then ***continue transactional execution***, wherein the processor handles the error at a later time (see par. [0093] of the instant application). In some embodiments, the processor handles the

instruction upon encountering a commit instruction at the end of the transaction (see par. [0093] of the instant application)

The FAIL instruction is distinct from the ABORT instruction in the combination of Moss and Oplinger because **the FAIL instruction provides the programmer with the ability to delay the processing of failures in certain cases**. In other words, the FAIL instruction can be used to record the occurrence of a condition which the processor can handle later (i.e., which does not require immediate handling). For example, if one or more operations are to be performed before the processor leaves transactional execution, the FAIL instruction can set a flag. The processor can then perform the operations before handling a subsequent commit instruction (see par. [0093] of the instant application).

Moss and Oplinger disclose an ABORT instruction that is confined to aborting transactional execution and branching to a location set by the TRY instruction. Nothing in Moss or Oplinger, alone or in concert suggests a FAIL instruction that sets state information within the processor to indicate that a failure has occurred.

Applicant has amended claims 1, 13, and 25 to clarify that the FAIL instruction sets state information within the processor to indicate that a failure has occurred, wherein the failure is handled at a later time. Applicant has also added new claims 26-28, which depend upon the independent claims to clarify that the later time can be when a commit instruction is encountered at the end of the transaction. The amendments find support in par. [0092] and [0093] of the instant application. No new matter has been added.

Hence, Applicant respectfully submits that the independent claims as presently amended are in condition for allowance. Applicant also submits that the dependent claims are in condition for allowance for the same reasons and for reasons of the unique combinations recited in such claims.

CONCLUSION

It is submitted that the application is presently in form for allowance. Such action is respectfully requested.

Respectfully submitted,

By __/Anthony Jones/_____
Anthony Jones
Registration No. 59,521

Date: 20 February 2008

Anthony Jones
Park, Vaughan & Fleming LLP
2820 Fifth Street
Davis, CA 95618-7759
Tel: (530) 759-1666
Fax: (530) 759-1665
Email: tony@parklegal.com